Exam - Take home

Table of contents


```r
# loading data
data_fin_exam <- readRDS('data/data_fin_exam.rds')
```

These two exercises are intentionally less directed than the in-class exam to encourage exploration and personal implementation. The goal of the exercises is to explore a method together with corresponding R packages, understand their basic usage then implement it or experiment their behaviour.

1 Exercise: Lasso: choice of lambda using hold out / cross-validation (20% total points)

TO DO at-home

The aim of this exercise is to find an optimal value of the lambda parameter for Lasso regression. You won't use the built-in functions `glmnetUtils::cv.glmnet` or `glmnet::cv.glmnet`, as done for the exam.

- First assume a simpler problem where your data set is split in training and testing set (you can use the exam `data_fin_exam`/`data_fin_holdout`).

```
# YOUR CODE HERE
```

- Fit the Lasso path (more details here) on the training set making use of `glmnetUtils::glmnet` or `glmnet::glmnet` functions

```
# YOUR CODE HERE
```

- using the `glmnet` object and `predict` function, for each lambda of the Lasso path (or better for all lambdas at once) obtain the predicted probabilities on the testing/holdout set

```
# YOUR CODE HERE
```

- using the preceding step you should be able to compute the hold-out error (or criterion if AUC) for each lambda (you can use a `for` loop), then conclude on the best lambda

```
# YOUR CODE HERE
```

- going one step further, replace the hold-out approach by a K-Fold Cross Validation (for each fold of you data, compute the cross-validation error (or criterion) for each lambda (you can use two imbricated `for` loops)); you can then conclude on the best lambda in terms of mean cross-validation error (or criterion). Finally compare the lambda chosen using your approach with the built-in function `cv.glmnet` described here.

```
# YOUR CODE HERE
```

2 Exercise: Decision trees (30% total points)

TO DO at-home

- `CART/rpart`
  - Using the data set `data_fin_exam`, build and plot a decision tree of you choice using `rpart`. You can use default parameters, and this way learning them. (see for example 2. Building the tree here

Try to display class probabilities and number of observations per node/leaf (for that explore functions `rpart.plot` or `pdp` from package `rpart.plot` see vignette).

```
# YOUR CODE HERE
```

- Playing with `rpart` parameters (in particular those inside `rpart.control`), fit, plot, then compare a "large/deep" decision tree and a "smaller/shallower" tree in terms of ROC/AUC/prediction on the holdout set.

```
# YOUR CODE HERE
```

- Describe in simple terms the output of the `printcp` function (see for example 4. Pruning the tree here. You don't have to understand deeply the pruning process but understand what is at stake in the process.

```
# YOUR CODE HERE
```

- Choose an arbitrary terminal number of leafs and prune the tree using the `prune` function.

```
# YOUR CODE HERE
```

- Select the optimal `cp` parameter for your tree (you can use a plot) and compare to the "large" and "small" models in terms of AUC.

```
# YOUR CODE HERE
```

- Gradient boosting

Using a gradient boosting package of your choice (`gbm`, `xgboost`), play with number of boosting iterations, weak learner complexity (decision tree depth, min number of observations), learning rate. Compare also logistic loss with exponential loss (adaboost) as was done in the lesson 3 of the course using a naive implementations.

```
# YOUR CODE HERE
```

You can take inspiration from this graph: here)

Example usage of `gbm` (more here, see for example the Figure 3 showing Out-of-sample predictive performance by number of iterations and shrinkage):

```
# library(gbm)
# gbm_deg <- gbm(Y~.,
#             data = YOUR_DATA,
#             n.trees = N, # number of boosting iterations
#             distribution = "bernoulli", # loss minimized
#             interaction.depth = 1,
#             /!\ interaction.depth (~#{terminal nodes}+1) <> rpart maxdepth
#                 n.minobsinnode = 1, # comparable to rpart minbucket
```

```
#            shrinkage = 1, # learning rate
#            bag.fraction = 1) # set at 1 to implement pure boosting
#            # (otherwise stochastic boosting, ie mix of boosting and bagging)
```

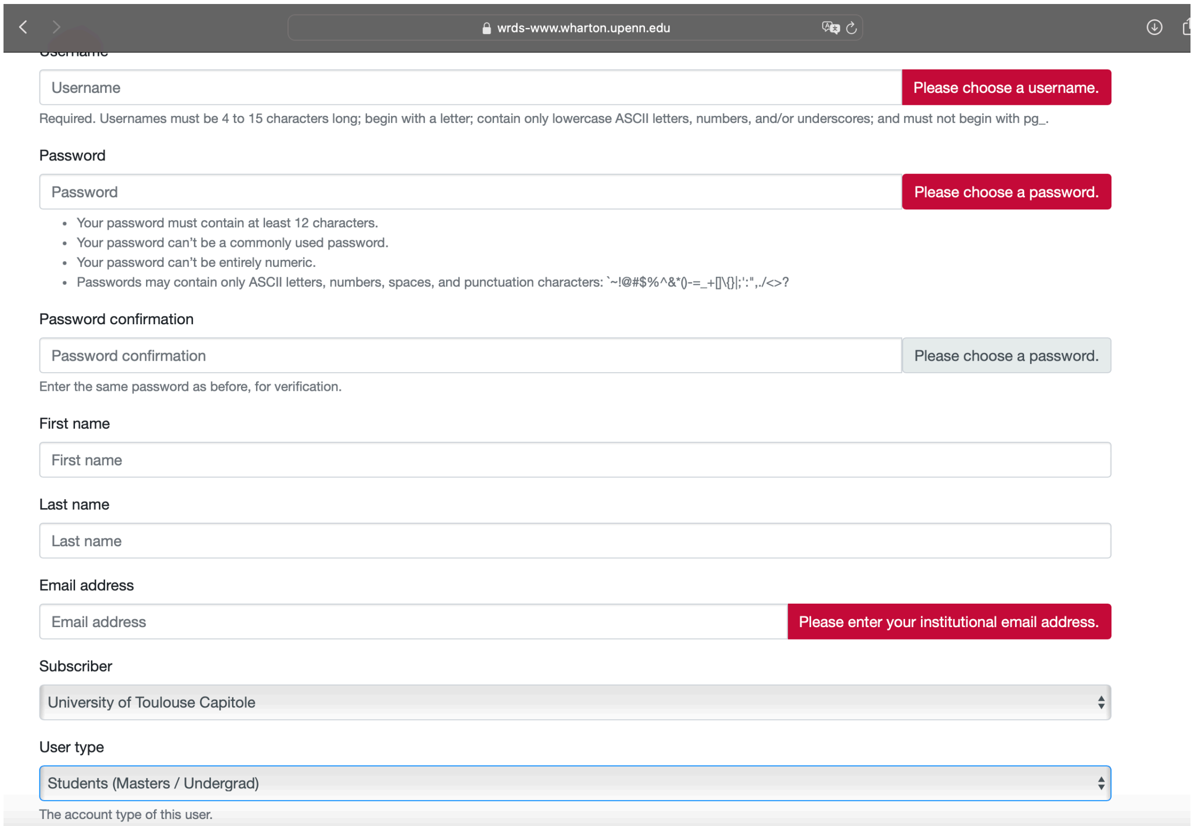
Example usage of `xgboost` (more here or here for the `R` package):

```
#library(xgboost)
# xgb_deg <- xgboost(data = as.matrix(YOUR_PREDICTORS),
#            label = as.vector(YOUR_TARGET),
#            max.depth = 1, # comparable to rpart maxdepth
#            eta = 1, # learning rate
#            nthread = 1,
#            nrounds = num_tree, # number of boosting iterations
#             min_child_weight = 0, # similar but not comparable to rpart minbucket
#             objective = "binary:logistic", # loss minimized
#             lambda = 0, # set at 0 to avoid L2 penalization
#             tree_method = "exact")
```

3 Project preparation - before next week lesson - Register to WRDS / Groups composition due 9 October (3 is good so that you can split tasks efficiently)

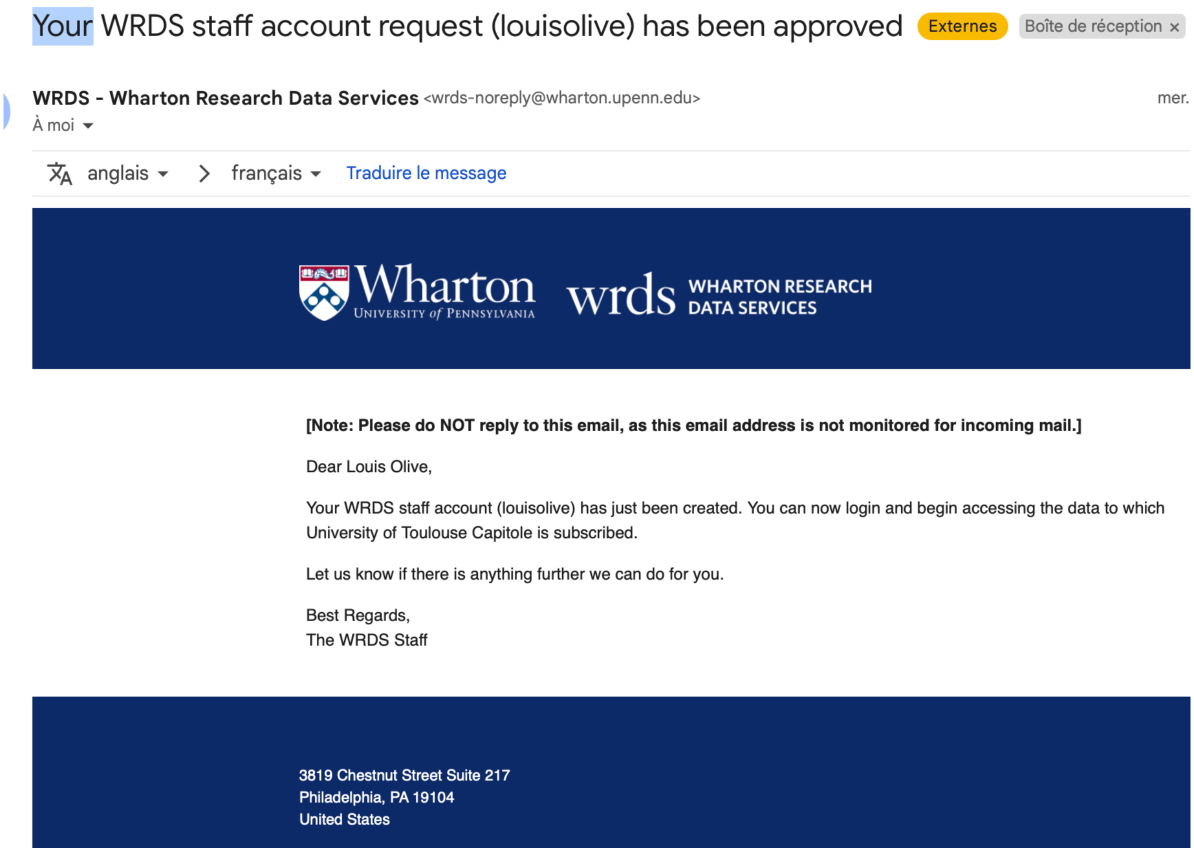
TO DO at-home

In order to access the project data, you have to register a `WRDS` account with your `@ut-capitole.fr` email. First go to this url: https://wrds-www.wharton.upenn.edu/register/ and follow the steps:



It takes roughly a week (or less) to be validated by `ut-capitole` teams.

You will get a user/password and will have to setup a 2FA authentication (preferably using the mobile app `Duo Mobile`).


Your WRDS staff account request (louisolive) has been approved Externes Boîte de réception ×

WRDS - Wharton Research Data Services <wrds-noreply@wharton.upenn.edu> mer.
À moi

anglais > français Traduire le message

Wharton University of Pennsylvania wrds WHARTON RESEARCH DATA SERVICES

[Note: Please do NOT reply to this email, as this email address is not monitored for incoming mail.]

Dear Louis Olive,

Your WRDS staff account (louisolive) has just been created. You can now login and begin accessing the data to which University of Toulouse Capitole is subscribed.

Let us know if there is anything further we can do for you.

Best Regards,
The WRDS Staff

3819 Chestnut Street Suite 217
Philadelphia, PA 19104
United States

Once you get your user/password, you can test an SQL request with `R` to the WRDS server as explained here in detail, or in the sample code below:

```
library(tidyverse)
library(dbplyr)
library(RPostgres)

# First create two environment variables to connect wrds
# in a terminal: touch $HOME/.Renviron
# inside the .Renviron file
# wrds_user = your_user
# wrds_password = your_password

wrds <- dbConnect(
    Postgres(),
    host = "wrds-pgdata.wharton.upenn.edu",
    dbname = "wrds",
    port = 9737,
```

```r
    sslmode = "require",
    user = Sys.getenv("wrds_user"),
    password = Sys.getenv("wrds_password")
)

# Otherwise use user/password within your code at your own risk
# wrds <- dbConnect(
#   Postgres(),
#   host = "wrds-pgdata.wharton.upenn.edu",
#   dbname = "wrds",
#   port = 9737,
#   sslmode = "require",
#   user = "YOUR_WRDS_USER",
#   password = "YOUR_WRDS_PWD"
# )


# Retrieve Altman ratios for APPLE INC

# Use dplyr verbs with a remote database table
# https://dbplyr.tidyverse.org/reference/tbl.src_dbi.html
funda_db <- tbl(wrds, in_schema("comp", "funda"))
funda_db %>%
  filter(grepl('APPLE INC', conm)) %>%
  select(gvkey, fyear, conm, at, wcap, re, ebit, lt, sale) %>%
  mutate(WCTA = wcap / at,
         RETA = re / at,
         EBTA = ebit / at,
         TLTA  = lt / at, # as a proxy for ME/TL
         SLTA = sale / at)
```

```
# Source:   SQL [?? x 14]
# Database: postgres  [louisolive@wrds-pgdata.wharton.upenn.edu:9737/wrds]
   gvkey  fyear conm       at   wcap    re  ebit    lt  sale  WCTA  RETA  EBTA
   <chr>  <int> <chr>   <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 001690  1980 APPLE~   65.4   16.3  14.5  23.6  39.4  117. 0.250 0.222 0.361
 2 001690  1981 APPLE~  255.   157.   54.1  66.1  77.5  335. 0.615 0.212 0.260
 3 001690  1982 APPLE~  358.   231.  116.  102.  101.   583. 0.647 0.324 0.286
 4 001690  1983 APPLE~  557.   340.  194.  130.  179.   983. 0.611 0.349 0.233
 5 001690  1984 APPLE~  789.   432.  256.   91.4 324.  1516. 0.548 0.324 0.116
 6 001690  1985 APPLE~  936.   527.  316.  147.  386.  1918. 0.563 0.337 0.157
 7 001690  1986 APPLE~ 1160.   712.  467.  274.  466.  1902. 0.614 0.403 0.236
```

```
 8 001690  1987 APPLE~ 1478.   829.   573.  371.   641.  2661. 0.561 0.387 0.251
 9 001690  1988 APPLE~ 2082.   956.   777.  620.  1079.  4071. 0.459 0.373 0.298
10 001690  1989 APPLE~ 2744.  1399.  1170.  634.  1258.  5284. 0.510 0.427 0.231
# i more rows
# i 2 more variables: TLTA <dbl>, SLTA <dbl>
```